1. **Test Cases**

The manual testing was a crucial tool to make sure that everything in the system works as intended and the tiers cooperated with each other the way they were intended. The tests were created based on use case descriptions. For each implemented description a test was done to make sure it is implemented correctly. For these tests we used black box testing.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario | Test scenario | Test steps | Test data | Expected results | Actual results | Pass/Fail |
| 1 | Login into the account | 1. The user enters his email. 2. The user enters his password |  | The user is logged in | As expected | Pass |
| 2 | Creating a user | 1. The user enters his credentials (First name, Last name, e-mail address, phone number, password, and gender) 2. User presses the register button |  | The user is registered in the system | As expected | Pass |
| 3 | View Marketplace | 1. The user presses the marketplace button. 2. The user will be redirected to the marketplace |  | The marketplace will show a list with all listened items | As expected | Pass |
| 4 | Filtering items | 1. The user can filter items by choosing the category, currency, and price range |  | The marketplace will show the items with filter | As expected | Pass |
| 5 | See post details | 1. The user presses the marketplace button 2. The user chooses the item he wants to buy |  | The program will show all the information about that item | As expected | Pass |
| 6 | Message seller | 1. The user presses the marketplace button 2. The user chooses the item he wants to buy 3. The user presses the message seller button 4. The user will be redirected to messages |  | The user can see the conversation of the item he wants to buy and to message the seller | As expected | Pass |
| 7 | Create post | 1. The user must login into his account 2. The user presses the create post button 3. The user enters all the necessary data 4. The user will be redirected to another window to insert a picture |  | The item will be placed on the marketplace | As expected | Pass |
| 8 | Edit post details | 1. The user must login into his account 2. The user goes to his items 3. The user chooses the item he wants to edit 4. The user presses the edit button 5. The user edits the fields he wants 6. The user submits the changes |  | The item will be edited with the new changes both on users’ profile and marketplace | As expected | Pass |
| 9 | Delete item | 1. The user must login into his account 2. The user goes to his items 3. The user chooses the item he wants to edit 4. The user presses the delete button |  | The item will be deleted from users profile and marketplace | As expected | Pass |
| 10 | Edit user | 1. The user must login into his account 2. The user presses the profile icon 3. The user chooses the information he wants to change 4. The user submits the changes |  | The new information about the user will be updated | As expected | Pass |
| 11 | Log out | 1. The user must login into his account 2. The user presses the profile icon 3. The user presses the log out button |  | The user will be logged out from his account | As expected | Pass |

**White box testing cases:**

**Presentation tier:**

This tier was tested with Bunit, xunit and Nunit.AutoFixture Libraries

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | Test Scenario | Test steps | Expected result | Actual result | Pass/Fail |
| 1 | User can be created | 1. Data required for registration is inputted 2. System calls create method for the user | An Id for the created user should be able to be checked | As expected | Pass |
| 2 | User should be able to login | 1. Data required for logging in is added 2. Authorisation is added 3. System calls LoginAsync method | Login page should be converted to Edit user page | As  expected | Pass |
| 3 | Item should be created | 1. Authorisation is added 2. Data required for item creation is inputted 3. The callouts are mocked 4. System calls create method for the item | An Id of the item should be able to be checked | As expected | Pass |

**Business logic tier:**

This tier was tested with xunit library, and the Data access tier has to be running for the tests to work

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | Test Scenario | Test steps | Expected result | Actual result | Pass/Fail |
| 1 | Create and get user | 1. Data needed for user creation is inputted. 2. System calls createUserAsync method 3. System calls getUsers method | The name of the created user matches the name of the user the system got | As expected | Pass |
| 2 | Create and get Item | 1. Data needed for item creation is inputted. 2. System calls createItemAsync method 3. System calls getItems method | The name of the created item matches the name of the user the system got | As expected | Pass |
| 3 | Edit profile | 1. Data needed for profile editing is inputted 2. System calls updateUserAsync method 3. System calls getUsers method | The name of the updated user matches the name of the user the system got | As expected | Pass |
| 4 | Delete user | 1. Id of the user that needs to be deleted is inputted 2. System calls deleteUserAsync method 3. System calls getUsers method 4. The system loops through existing users to check if the user with the matching id still exists | The user with the deleted id should not be in the database | As expected | Pass |
| 5 | Edit Item | 1. Data needed for Item editing is inputted 2. System calls updateItemAsync method 3. System calls getItemsByIdAsync method | The name of the updated Item matches the name of the Item the system got | As expected | Pass |
| 6 | Delete Item | 1. Id of the Item that needs to be deleted is inputted 2. System calls deleteItemAsync method 3. System calls getItems method 4. The system loops through existing items to check if the item with the matching id still exists | The item with the deleted id should not be in the database | As expected | Pass |